Towards TCS Concepts for Characterizing Expertise in Learning Systems Validation (Algorithms and Theory of Computing)

AUTHOR(S):

Grieser, Gunter; Jantke, Klaus P.; Lange, Steffen

数理解析研究所講究録
1041 巻 1998 年 103-110

Towards TCS Concepts for Characterizing Expertise in Learning Systems Validation*

Gunter Grieser[†§] Klaus P. Jantke[†] Steffen Lange[‡¶]

Abstract

The authors' intention is to motivate the needs for the introduction of deeper formal concepts of Theoretical Computer Science (TCS) into particular Artificial Intelligence (AI) research.

The target application area of complex AI systems validation is briefly introduced and a key question is raised. It is the authors' ultimate goal to exemplify the relevance of TCS concepts for answering certain questions of the type under consideration.

An approach is exemplified in the particular area of inductive inference of recursive functions. Interactive scenarios of validating inductive inference algorithms are formalized.

Based on these TCS concepts, the paper is focused on the problem of characterizing the expertise which is necessary and sufficient in the validation of inductive inference systems, respectively. The crucial TCS concepts invoked are BLUM complexity measures, limiting and relativized computability.

1 Where to Apply TCS Concepts: Validation of Complex Systems – Necessity, Problems, and Solutions

There is an obvious necessity to validate and verify complex systems. It might easily happen that *...the inability to adequately evaluate systems may become the limiting factor in our ability to employ systems that our technology and knowledge will allow us to design.* (cf. [WW93])

Unfortunately, there are numerous severe accidents bearing abundant evidence for the truly urgent need for complex systems validation. Besides spectacular cases, daily experience with more or less invalid systems is providing paramount illustrative examples. Progress in the area of validation and verification of complex systems requires both disciplinary results and solutions in the humanities including cognitive psychology, e.g. Even social and political aspects come into play. The authors refrain from an in-depth discussion.

Following [Bo84] and [OO93], validation is distinguished from verification by the illustrative circumscription of dealing with *building the right system*, whereas verification deals with *building the system right.* The prototypical application area considered in the present paper is systems *validation*, which – according to the perspective cited above – is less constrained and less formalized than verification.

Assume computer systems which are designed and implemented for an interactive use to assist human beings in open loops of human–machine interactions of a usually unforeseeable length. The validation task is substantially complicated, if it is intermediately undecidable whether or not some human–machine cooperation will eventually succeed.

Nontrivial learning problems, for instance, are quite typical representatives of such a class of problems attacked through complex and usually time consuming sequences of human–machine interactions. Knowledge discovery in data bases, for instance, is a practically relevant application domain for those learning approaches.

For assessing those systems' validity, there have been proposed validation scenarios of several types (cf. [KPG97], e.g.). As soon as human experts are involved in the implementation of validation scenarios, there arises the problem of the experts' competence. An in-depth investi-

*A full version of this paper appeared as technical report MEME–MMM–1–98.

[†]Meme Media Laboratory, Hokkaido University, Sapporo 060, Japan, [grieser,jantke]@meme.hokudai.ac.jp

[‡]Universität Leipzig, Institut für Informatik, PF 920, 04009 Leipzig, Germany, slange@informatik.uni-leipzig.de

[§]This research has been supported by the German Research Fund (DFG) under contract number Ja 566/10-1.

[¶]This work was performed while the author was visiting the Meme Media Laboratory at Hokkaido University.

gation of validation scenarios, of their appropriateness for certain classes of target systems, and of their power and limitations involves inevitably reasoning about the experts' competence.

It is a **basic question** how to formalize interactive systems validation. Formal concepts are highly desirable for setting the stage for the derivation of justified answers to certain questions which are controversially discussed. Still informally speaking, the **key question** is *how to characterize the human expertise necessary or sufficient for validating certain AI systems.*

The issue of human expertise is usually understood a problem of cognitive sciences (cf. [Co92]). This is complicating a thorough computer science investigation of validation scenarios mostly based on formal concepts and methodologies.

Therefore, the present papers is focusing on approaches to characterize human expertise in formal terms. This is deemed a substantial step towards a better understanding of the power and limitations of interactive validation scenarios.

The **key question** is exemplified in the particular area of validating learning systems. *How to characterize the human expertise necessary or sufficient for validating systems which learn inductively ?*

The usage of TCS concepts demonstrates in formal terms that certain human expertise sufficient to accomplish certain validation tasks is substantially non-recursive. This leads to the interesting question whether or not humans are *provably* more powerful than machines – a question which we are not going to deal with.

2 Learning Systems Validation – A Case for Determining Expertise by means of Suitable TCS Concepts

We adopt validation scenarios according to [KPG97], e.g. Human experts who are invoked for learning systems validation within the framework of those scenarios need to have some topical competence. It is one of the key problems of validation approaches based on human expertise how to characterize the experts' potentials which allow them to do their job sufficiently well. Even more exciting, it is usually unknown whether or not the humans engaged in those interactive scenarios can be replaced by computer programs without any substantial loss of validation power. This problem is of a great philosophical interest and of a tremendous practical importance.

For learning systems validation, we will be able to characterize the human expertise *sufficient* for trustable systems validation. Some characterizations are even both *sufficient* and *necessary.*

2.1 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers, and let $\mathbb{N}_\perp = \mathbb{N} \cup \{\perp\}$. For any $M \subseteq \mathbb{N}$ we denote the power set of M by $\wp(M)$. For some function f, $dom(f)$ denotes the domain of f.

Computable functions are defined over $\mathbb{N}$. $\mathcal{P}$ is the class of all partial recursive functions. The class of total recursive functions is denoted by $\mathcal{R}$.

By $cod : \mathbb{N}^2 \to \mathbb{N}$ let us denote CANTOR's pairing function, i.e. an easy to compute function that is one to one, and onto (cf. [Ro67]).

For a GÖDEL numbering φ, each number $j \in \mathbb{N}$ is specifying a particular function denoted by φ_j. For the rest of this paper, a GÖDEL numbering φ (cf. [Ro67]) and a corresponding BLUM complexity measure ϕ (cf. [Bl67]) are fixed. For any $j, x \in \mathbb{N}$, $\varphi_j(x)\downarrow$ indicates that $\varphi_j(x)$ is defined.

For a set F of computable functions, the index set I_F contains exactly all programs for functions from F, i.e. $I_F = \{i \in \mathbb{N} \mid \varphi_i \in F\}$.

Let $U \subseteq \mathcal{R}$. Then, U is is said to be enumerable provided there is a $g \in \mathcal{R}$ such that $U \subseteq \{\varphi_{g(n)} \mid n \in \mathbb{N}\} \subseteq \mathcal{R}$. If $U = \{\varphi_{g(n)} \mid n \in \mathbb{N}\}$ for some $g \in \mathcal{R}$, then U is called exactly enumerable. By NUM (NUM!) we denote the collection of all enumerable (exactly enumerable) subsets of $\mathcal{R}$.

A sequence $(n_t)_{t \in \mathbb{N}}$ of natural numbers is said to converge to some ultimately final value n, if past some point t' all numbers n_t $(t \geq t')$ are identical to n. This is denoted by $\lim(n_t)_{t \in \mathbb{N}} = n$.

f is said to be limiting computable, if there is some $g \in \mathcal{R}^2$ satisfying (i) for all $x \in dom(f)$, there is some $t' \in \mathbb{N}$ such that, for all $t \in \mathbb{N}$ with $t > t'$, $g(x,t) = f(x)$, and (ii) for all $x \notin dom(f)$ and all $t \in \mathbb{N}$, there exists some $t' \in \mathbb{N}$ such that $t' > t$, and $g(x,t') \neq g(x,t)$.

Any $M \subseteq \mathbb{N}$ is said to be limiting decidable, if M's characteristic function χ_M is limiting computable. Similarly, $M \subseteq \mathbb{N}$ is said to be limiting enumerable, if 'half' of M's characteristic function χ_M^+ is limiting computable, where $\chi_M^+(x) = 1$ if and only if $x \in M$.

We use the abbreviation f^A to indicate that f is computable relative to some oracle A (cf. [Ro67]), i.e. there is an algorithm computing f that is allowed to ask, from time to time, question of the type "$n \in A$?", and that may use the answers supplied to determine how to continue.

We use the abbreviation $[M\,l.r.e.]^A$ to indicate that there is some A-computable function limiting enumerating set $M \subseteq \mathbb{N}$.

2.2 Inductive Inference – Notions and Notations

This section is focused on essential features of inductive learning which complicate the validation task, and it introduces a few basic formalisms. For both conceptual simplicity and expressive generality, the focus of the present investigations is on learning of total recursive functions from finite sets of input/output examples (cf. [AS83]).

When learning any total recursive function f, the input/output examples $\langle 0, f(0)\rangle$, $\langle 1, f(1)\rangle$, $\langle 2, f(2)\rangle$, ... are subsequently presented. Learning devices are computable procedures generating hypotheses upon natural numbers $f[t]$ encoding finite samples $\langle 0, f(0)\rangle, \langle 1, f(1)\rangle, \ldots, \langle t, f(t)\rangle$. Note that, for every $x \in \mathbb{N}$, there is the one and only finite sample encoded by x.

For notational convenience, hypotheses are just natural numbers which are to be interpreted via the underlying GÖDEL numbering φ. Note that learning will usually take place over time. Thus, hypotheses are generated subsequently.

An individual learning problem is always understood to be a class of target functions. A corresponding learning device has to learn each of these functions individually when fed with appropriate samples.

Definition 1 (LIM)
$U \in LIM$ if and only if there is an $S \in \mathcal{P}$ satisfying for any $f \in U$: (1) for all $t \in \mathbb{N}$, $h_t = S(f[t])$ is defined and (2) $\lim(h_t)_{t\in\mathbb{N}} = h$ with $\varphi_h = f$ exists.

Thus, LIM is a collection of function classes U for which some recursive learning device S as indicated exists. If the learning device S exclusively outputs indices for total recursive functions, then U belongs to the learning type TOTAL.

Definition 2 (TOTAL)
$U \in TOTAL$ if and only if there exists some $S \in \mathcal{P}$ satisfying for any $f \in U$: (1) for all $t \in \mathbb{N}$, $h_t = S(f[t])$ is defined, (2) $\lim(h_t)_{t\in\mathbb{N}} = h$ with $\varphi_h = f$ exists, and (3) for all $t \in \mathbb{N}$, $h_t \in I_R$.

Alternatively, if it is decidable whether or not S, when learning any $f \in U$, has reached the ultimate learning goal then S witnesses that U belongs to the special learning type FIN. This approach is easily formalized as well:

Definition 3 (FIN)
$U \in FIN$ if and only if there exist some $S \in \mathcal{P}$ and some related decision procedure $d \in \mathcal{P}$ which satisfy for any $f \in U$: (1) for all $t \in \mathbb{N}$, $h_t = S(f[t])$ is defined, (2) $\lim(h_t)_{t\in\mathbb{N}} = h$ with $\varphi_h = f$ exists, (3) for all $t \in \mathbb{N}$, $d(f[t])$ is defined, and (4) $d(f[t]) = 1$ if and only if $S(f[t]) = h$.

The relation between the learning types introduced above is as follows:

$$\text{FIN} \subset \text{TOTAL} \subset \text{LIM} \subset \wp(\mathcal{R}).$$

To sum up, although inductive learning succeeds after finitely many steps, in its right perspective, it is appropriately understood as a limiting process. This fact is causing unavoidable difficulties to validation attempts based on local information, only.

2.3 Interactive Scenarios for Learning Systems Validation

A validation problem for inductive inference systems is given as a triple of (1) some function class $U \subseteq \mathcal{R}$, (2) some learning function $S \in \mathcal{P}$, and (3) an inductive inference type like LIM, TOTAL, or FIN, e.g. The precise question is whether S is able to learn all functions f from U with respect to the considered inductive inference type ID. Naturally, this question is only worth to be asked in case that $U \in \text{ID}$.

There are two substantial difficulties. First, function classes U under consideration are usually infinite. Second, every individual function is an infinite object in its own right. In contrast, every human attempt to validate some learning system by a series of *systematic tests* is essentially finite. Thus, validity statements are necessarily approximate.

When some process of (hopefully) learning some target function $f \in U$ by some device $S \in \mathcal{P}$ with respect to some inductive inference type is under progress, one may inspect snapshots determined by any point t in time.

Any pair of an index of a recursive function and a time point is called *test data*. They represent initial segments of functions. Certain data are chosen for testing by a *test data selection*.

Definition 4 (Test Data, Test Data Selection) *Any pair $\langle j,t \rangle$ with $\varphi_j(x)\downarrow$, for all $x \leq t$, is called test data. TD denotes the set of all potential test data. Furthermore, a function $DS : \mathbb{N} \rightarrow \wp(TD)$ defines a test data selection provided that, for all $n \in \mathbb{N}$, $DS(n) \subset DS(n+1)$.*

In practice, the selection of test data is frequently done by hand. So, there is no need to consider the test data selection to be recursive.

Intuitively, the two numbers refer to a program and an intensity, with which the behaviour of the system is tested for this program. Subsequently, test data $\langle j,t \rangle$ are interpreted as $\varphi_j[t]$. Therefore, the second parameter is called a time stamp.

In order to verify whether or not a learning system is valid with respect to some function class U, enough relevant test data have to be selected.

Definition 5 (Completeness)
Let $U \subseteq \mathcal{R}$ and let DS be a test data selection. DS is said to be complete for U if and only if the set $T = \bigcup_{n \in \mathbb{N}} DS(n)$ satisfies conditions (1) for all $f \in U$, there is a φ-index j for f such that $\langle j,t \rangle \in T$ for every $t \in \mathbb{N}$, (2) there are only finitely many test data $\langle j,t \rangle \in T$ with $j \notin I_U$, and (3) there are only finitely many test data $\langle j,t \rangle \in T$ with $\langle j,t+1 \rangle \notin T$.

When testing some learning system S on test data $\langle j,t \rangle$, one is interested in knowing how S behaves on input $\varphi_j[t]$. *Experimentation* means feeding test data to the system under investigation and, if possible, receiving system's response.

Definition 6 (Experimentation)
Any mapping $Exp : \mathbb{N} \rightarrow \mathbb{N}_\perp$ is called experimentation. Furthermore, the mapping Exp is an experimentation for $S \in \mathcal{P}$ if and only if for all $\langle j,t \rangle \in TD$, either $Exp(\varphi_j[t]) = \perp$ or $Exp(\varphi_j[t]) = S(\varphi_j[t])$.

Because experimentation is a human activity, the mapping Exp is not necessarily computable.

Intuitively, the result $\perp$ means that no proper system's response has been received. This may be due to some time out, e.g. Clearly, if it frequently happens that $Exp(\varphi_j[t]) = \perp$, but $S(\varphi_j[t])$ terminates, then this particular experimentation does not reflect the learning system's behaviour sufficiently well.

Insistency characterizes a manner of interactively validating a system where the human interrogator does never give up too early.

Definition 7 (Insistency)
Let Exp be an experimentation for $S \in \mathcal{P}$. Exp is said to be insistent for S if and only if $Exp(\varphi_j[t]) = S(\varphi_j[t])$ for exactly all $\langle j,t \rangle \in TD$, where $S(\varphi_j[t])\downarrow$.

These formalisms are aimed at the description of an expert's interactive validation of any given learning system S. An expert is performing experiments with some target object φ_j in mind resulting in *protocols*. A protocol is a triple $\langle j,t,h \rangle$ with $\langle j,t \rangle \in TD$ and $h = Exp(\varphi_j[t])$.

Those protocols are subject to the expert's evaluation marked 1 or 0, respectively, expressing the opinion whether or not the experiment witnesses the system's ability to learn the target function φ_j. This realizes a certain mapping $Eval : TD \times \mathbb{N} \rightarrow \{0,1\}$, a so-called expert's evaluation function. As before, this might be not computable. The tuple consisting of a protocol and the expert's evaluation is a *report*.

Validation statements are synthesized upon reports which reflect interactive systems validation to some extent. In dependence on the underlying validation scenario, there are concepts of different sophistication. We adopt the most simple approach, and consider any finite set of reports to be a *validation statement*.

For interactive systems, in general, and for learning systems, in particular, any one–shot validation does not seem to be appropriate. Thus, one is lead to validation scenarios in open loops which result in sequences of validation statements. Hence, a *validation dialogue* arises, constituted by any test data selection, experimentation, and the expert's evaluation function.

Definition 8 (Validation Dialogue)
Assume any test data selection DS, any experimentation Exp, and any expert's evaluation function Eval. The triple $VD = \langle DS, Exp, Eval \rangle$ *defines a sequence of validation statements* $(VS_n)_{n \in \mathbb{N}}$ *called a validation dialogue, where, for all* $n \in \mathbb{N}$, VS_n *is the collection of all reports* $\langle \langle j,t,h \rangle, b \rangle$ *with* $\langle j,t \rangle \in DS(n)$, $h = Exp(\varphi_j[t])$, *and* $b = Eval(j,t,h)$.

Such a validation dialogue is said to be *successful* for U and S if and only if the underlying data selection is complete for U, the experimentation is insistent, and the experts' evaluation is converging to the success value 1, for every program which is subject to unbounded experimentation.

Definition 9 (Successful Validation Dialogue) *Assume* $U \subseteq \mathcal{R}$, $S \in \mathcal{P}$, *any test data selection DS, any experimentation Exp, and an expert's evaluation function Eval. The validation dialogue* $(VS_n)_{n \in \mathbb{N}}$ *defined by* $VD = \langle DS, Exp, Eval \rangle$ *is successful for U and S if and only if (1) DS is complete for U, (2) Exp is insistent for S, and (3) for every* $j \in \mathbb{N}$, *there are only finitely many reports* $\langle \langle j,t,h \rangle, b \rangle \in \bigcup_{n \in \mathbb{N}} VS_n$ *with* $b = 0$.

The formal concepts introduced will suffice for systematically investigating the possibilities of interactive learning systems validation.

3 Characterizing Test Data Selection: The Goal of Invoking TCS Concepts

We go only very briefly into the details of test data selection. There are several areas of more traditional computer science and of AI where the generation of test cases or test sets plays an important role. The methodologies invoked range from sophisticated mathematical considerations to comprehensive investigations taking aspects of cognitive psychology into account. We are aware of the narrowness of our present approach, but we had to trade generality for precision.

Definition 10 (CDS)
Let $U \subseteq \mathcal{R}$. $U \in CDS$ *if and only if there is a computable test data selection being complete for U.*

Let A be an oracle, and let $U \subseteq \mathcal{R}$. Then, we let $[U \in \mathrm{CDS}]^A$ indicate that there is an A-computable data selection being complete for U.

In most investigations, it is of a particular interest to find out whether or not those sets of test cases can be generated automatically. Within the technical terms of the present approach, this is the question for the computability or relativized computability of the data selection DS.

Theorem 1 *Let* $U \subseteq \mathcal{R}$ *and A be any oracle. Then, it holds:* $[U \in CDS]^A \Longleftrightarrow [U \in NUM!]^A$.

This allows the following corollary exhibiting the restrictiveness of areas in which the selection of relevant test data can be fully automated.

Corollary 2 $CDS = NUM!$.

It can be shown, that there is no level of expertise that allows to generate complete test data for all learning problems. More formally:

Theorem 3 *Let A be any oracle. Then, there is some* $U \in LIM$ *such that* $[U \notin CDS]^A$.

4 Characterizing Experiment Control: The Goal of Invoking TCS Concepts

The question considered in the present section is how to implement any form of control to accomplish insistent experimentation. Conceptually, one needs any module supervising experimentation and "telling" the validator whether or not (s)he should wait a little longer for some system's response.

TCS concepts are required to make this intuitive approach precise. Otherwise, there were no hope for precise results about the possibilities of implementing insistent experimentation. The following seems to be rather straightforward.

Any given learning function S under validation is effectively computable and, therefore, when being subject to experimentation, may be understood as some particular φ_s with $s \in \mathbb{N}$. Note that this does not mean that the validator is necessarily aware of the particular "program" s under inspection. However, the actual experimentation process is characterized by the computation time of φ_s which can be suitably formalized by the related BLUM complexity measure.

In order to formalize control concepts of insistent experimentation, it is necessary to distinguish between so-called 'white box' validation and 'black box' validation (cf. [Gu93] and

[GR97], e.g.). In the first case, one has access to the program under validation, whereas one is restricted to only the program's behaviour, in the latter case. This is formally reflected by a control function c which depends either on both the information $\varphi_j[t]$ presented and the program s inspected or on the recent information $\varphi_j[t]$, only.

Definition 11 (Control)
Let $c \in \mathcal{R}^2$, and let $s \in I\!N$. Then, c allows for an insistent white box experimentation with φ-program s if and only if, for any $\varphi_j[t] \in I\!N$, $\varphi_s(\varphi_j[t])\downarrow$ implies $c(\varphi_j[t],s) \geq \phi_s(\varphi_j[t])$.

Let $c \in \mathcal{R}$, and let $s \in I\!N$. Then, c allows for an insistent black box experimentation with φ-program s if and only if, for any $\varphi_j[t] \in I\!N$, $\varphi_s(\varphi_j[t])\downarrow$ implies $c(\varphi_j[t]) \geq \phi_s(\varphi_j[t])$.

Let $c \in \mathcal{R}^2$. Then, $COP^w(c)$ is the set of all φ-programs controlled by c. Furthermore, $COF^w(c)$ is the set of all computable functions for which there is a φ-program controlled by c, i.e. $COF^w(c) = \{\varphi_s \mid s \in COP^w(c)\}$. Concerning insistent black box experimentation via some control $c \in \mathcal{R}$, the sets $COP^b(c)$ and $COF^b(c)$ are defined analogously.

As the reader may easily verify, insistent experimentation for any single strategy can be implemented by an insistent control, and vice versa. In the following, we study to what extent insistent control functions can be used to realize experiments with arbitrary learning devices.

It is well-known that there are arbitrary complex programs. In other words, for each recursive bound[1], there are infinitely many total recursive functions that have a program which exceeds this bound (cf. [Bl67]). Thus, one may expect that insistent experimentation for larger classes of inductive learning devices requires some non-recursive expertise.

A prominent example for non-recursive expertise is the halting set K, i.e. $K = \{\langle i,x\rangle \mid i,x \in I\!N, \varphi_i(x)\downarrow\}$. As we will see, the halting set K exactly characterizes the level of non-recursive expertise which is both necessary and sufficient to realize white box experimentation for all learning devices.

[1] A total function $c \in \mathcal{R}$ is said to be a complexity bound for the φ-program j of some $f \in \mathcal{R}$ provided that $c(x) \geq \phi_j(x)$ for almost all $x \in I\!N$.

Let A be an oracle, let $\mathcal{S} \subseteq \mathcal{P}$, and let $Q \subseteq I\!N$. Then, we use the notations $[\mathcal{S} \in COP^w]^A$ and $[Q \in COF^w]^A$ to indicate that there is an A-computable control c^A which allows for an insistent white box experimentation with all programs in Q and all learning devices in $\mathcal{S}$, respectively. We adopt these notations for black box experimentation.

Theorem 4 *Let A be any oracle. Then, we have: $[I_{\mathcal{P}} \in COP^w]^A \Longleftrightarrow [K \text{ is recursive}]^A$.*

In the black box approach, the situation changes drastically. In this setting, the control c does not receive any information about the program it is supposed to control. Thus, the result from [Bl67] already mentioned above immediately allows for the following insight.

Theorem 5 *There is no oracle A that guarantees $[\mathcal{P} \in COF^b]^A$.*

5 Characterizing Validation Expertise: The Goal of Invoking TCS Concepts

Within the last two sections, we have investigated the problem of automatizing the data selection and the experimentation. Now, we focus our attention on the evaluation phase. The next definition provides the formal framework for an appropriate investigation.

Definition 12 (EVAL)
Let ID be an identification type, and A be an oracle. Then, all learning devices are said to be ID-evaluable with respect to all learning problems modulo oracle A ($[ID \in EVAL]^A$, for short) if and only if there is an A-computable evaluation $Eval^A$ such that, for all $S \in \mathcal{P}$, all $U \in ID$, all data selections DS complete for U, and all experimentations Exp insistent for S, we have: the validation dialogue $VD = \langle DS, Exp, Eval^A\rangle$ is successful for U and S if and only if $U \subseteq ID(S)$.

The key question considered here is, given some learning type ID, how powerful an expert must be to ID-evaluate *all* computable learning devices with respect to all learning problems that can be solved by learning devices that meet the requirements of the learning type ID.

TCS concepts can be invoked to implement the following program:

(i) Choose the requirements an acceptable learning device should meet. For instance, it should learn in the limit or should finitely learn.

(ii) Find some characterization of expertise.

(iii) Prove a theorem that any expert who is competent according to the conditions of (ii) is, therefore, able to evaluate whether or not any given learning device meets the requirements focused under (i) when being confronted with some learning problem.

(iv) Prove a theorem that an expert's ability to evaluate all devices with respect to the requirements fixed within (i) necessarily needs some skill as formalized within (ii).

Imagine, for a moment, some validation dialogue for a LIM-learning device. The expert, who is involved in this process, has eventually to answer the following questions:

(a) Does the learner, when fed information for some function f, always output a hypothesis?

(b) Does the learning device converge when successively fed information about f?

(c) Does the final hypothesis correctly describe the target function f?

When asking for experts' expertise for validation of machines which have to meet further requirements, like for TOTAL- or FIN-type learners, e.g., answering questions (a), (b), and (c) is an essential part of the evaluation process as well. Therefore, we first try to characterize the expertise needed to answer them.

Because the experimentation is assumed to be insistent, (a) and (b) are not difficult to answer. (c) is in essence the crucial question in this section. If one can determine in the limit, at least, whether a hypothesis is an index for a total computable function, the consistency check becomes fairly simple. Without this additional knowledge, it might be impossible to test consistency.

So, the problem of determining in the limit, whether or not an arbitrary computable function is total, seems to play a key role in validating learning systems.

Proposition 6 *Let A be any oracle. Then, we have the following equivalences:*
(1) $[LIM \in EVAL]^A \iff [I_R \ l.r.e.]^A$.
(2) $[TOTAL \in EVAL]^A \iff [I_R \ l.r.e.]^A$.
(3) $[FIN \in EVAL]^A \iff [I_R \ l.r.e.]^A$.

Proposition 6 characterizes validation expertise. Next, we relate validation expertise to domain expertise, i.e. the ability to solve learning problems in the required sense.

Proposition 7 ([AB91]) *Let A be any oracle. Then, we have:* $[I_R \ l.r.e.]^A \iff [\mathcal{R} \in LIM]^A$.

Putting the last two results together, we immediately arrive at the following insight.

Theorem 8 *Let A be any oracle. Then, we have:* $[LIM \in EVAL]^A \iff [\mathcal{R} \in LIM]^A$.

Consequently, an expert having the ability to LIM-evaluate all learning devices has a level of expertise which is sufficient to solve every possible learning task, i.e. to learn in the limit every $f \in \mathcal{R}$ from input/output examples.

Finally, we summarize the insights obtained.

Theorem 9 *Let A be any oracle. Then, the following statements are equivalent:*
(1) $[LIM \in EVAL]^A$, *(2)* $[TOTAL \in EVAL]^A$,
(3) $[FIN \in EVAL]^A$, *and (4)* $[\mathcal{R} \in LIM]^A$.

Although different types of learning behaviour may need different approaches in validating it, the required expertise remains unchanged.

6 Conclusions

First, we sum up very briefly the technical contents of the present paper. We know about *sufficient* and *necessary expertise* to accomplish some validation tasks. Interestingly, this expertise *can not be automated.* The strength of the expertise is illustrated by the following informal statement:

Who is able to validate certain learning devices, is also able to replace them in solving learning problems.

Evidence for this thesis is provided by several of our results above. There are some results illuminating the necessity to have an expertise formally expressed by the oracle K, i.e. by the power to decide the halting problem. In case this power is available, it is immediately possible to learn in the limit any recursive function: $[\mathcal{R} \in \mathrm{LIM}]^K$.

Some of our results exhibit non-recursive expertise as a necessary prerequisite to accomplish certain tasks in learning systems validation.

Consequently, there is no way to replace humans by computer programs for those validation tasks. Does this mean that we can prove that humans are more powerful than machines in this particular area? We don't know! In case humans turn out to be able to solve all those validation problems sufficiently well, this were some evidence for the humans' superiority to machines – an exciting open question.

However, these remarks refer only to the technical perspective of our present paper. Our starting point was more general.

The validation of complex systems is a remarkably urgent problem area. Several validation approaches and scenarios are recently under development, under theoretical investigation, and also under experimental exploration.

As soon as human experts are becoming involved, the question for the experts' competence is becoming crucial. Most problems, even some very fundamental one, are still open. A quite typical question is how the validation experts' expertise relates to the domain experts' skills. Is it necessary that anybody involved in systems validation needs to be qualified for doing the systems' job, at least in principle? Or does a substantially lower degree of qualification suffice for validating a system's behaviour?

There might be no generally valid answers to those questions. Despite this, any clear answer derived under certain more specific circumstances might be discussed controversially. Therefore, a firm justification is both theoretically and practically relevant. Unwelcome findings need a particularly serious support.

TCS concepts and methodologies may provide a firm basis for sufficiently clear statements which are characterizing human expertise for complex systems validation. The present paper is intended to provide an example, only. There is evidence for the thesis that validation is not simpler than doing the job itself.

References

[AB91] L. M. Adleman and M. Blum. Inductive inference and unsolvability. *The Journal of Symbolic Logic*, 56(3):891–900, Sept. 1991.

[AS83] D. Angluin and C. H. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[Bl67] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM*, 14:322–336, 1967.

[Bo84] B. W. Boehm. Verifying and validating software requirements and design specifications. *IEEE Trans. Software*, 1(1):75–88, 1984.

[Co92] N. J. Cooke. Modeling human epertise in expert systems. In Hoffman (ed.), *The Psychology of Expertise. Cognitive Research of Empirical AI*, pp. 29–60. Springer-Verlag, 1992.

[GJL98] G. Grieser, K. P. Jantke, and S. Lange. Characterizing sufficient expertise for learning system validation. In *Proc. Florida AI Research Symp. '98*. AAAI Press, 1998. to appear.

[GR97] A. J. Gonzalez and P. Ramasamy. Detecting anomalies in constraint-based systems. In Gens (ed.), *Int. Sci. Colloquium, Ilmenau Univ. of Technology*, vol. 2, pp. 35–40. TU Ilmenau, 1997.

[Gu93] U. G. Gupta. *Validation and Verification of Expert Systems*. IEEE Press, Los Alamitos, CA, 1993.

[KPG97] R. Knauf, I. Philippow, and A. J. Gonzalez. Towards an assessment of an AI system's validity by a TURING test. In Dankel II (ed.), *Proc. Florida AI Research Symp. '97*, pp. 397–401. Florida AI Research Society, 1997.

[OO93] R. M. O'Keefe and D. E. O'Leary. Expert system verification and validation: A survey and tutorial. *Artificial Intelligence Review*, 7:3–42, 1993.

[Ro67] H. Rogers jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[WW93] J. A. Wise and M. A. Wise. Basic considerations in verification and validation. In Wise, Hopkin, and Stager (eds.), *Verification and Validation of Complex Systems: Human Factors Issues, NATO ASI Series, Series F: Comp. and Systems Sci.*, vol. 110, pp. 87–95. Springer–Verlag, 1993.